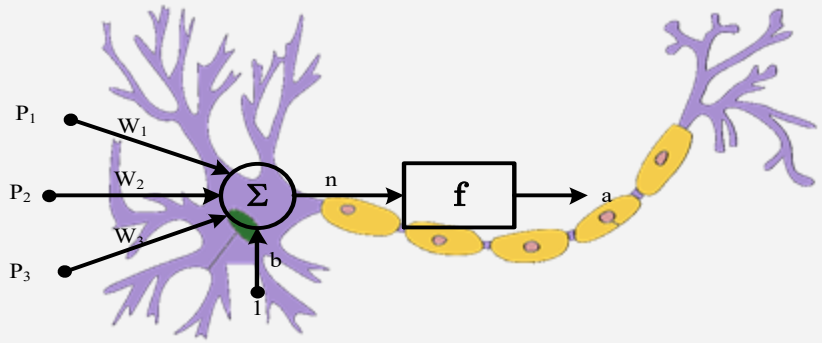
Luiz Gustavo Lourenço Moura
luiz.gustavo@gsuite.iff.edu.br

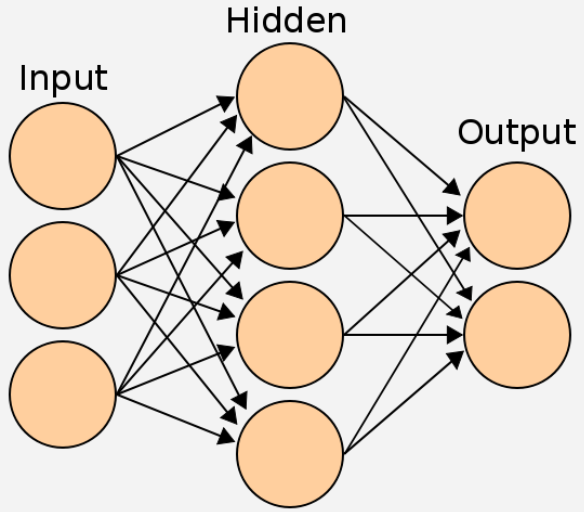
Redes neurais convolucionais
(Convolutional Neural Networks (CNNs / ConvNets)

REDES NEURAIS

- O neurônio coleta sinais do canal de entrada (dendritos), processa a informação no núcleo e gera uma saída pelo axônio
- O aprendizado humano ocorre adaptativamente por meio da variação da força de ligação entre os neurônios

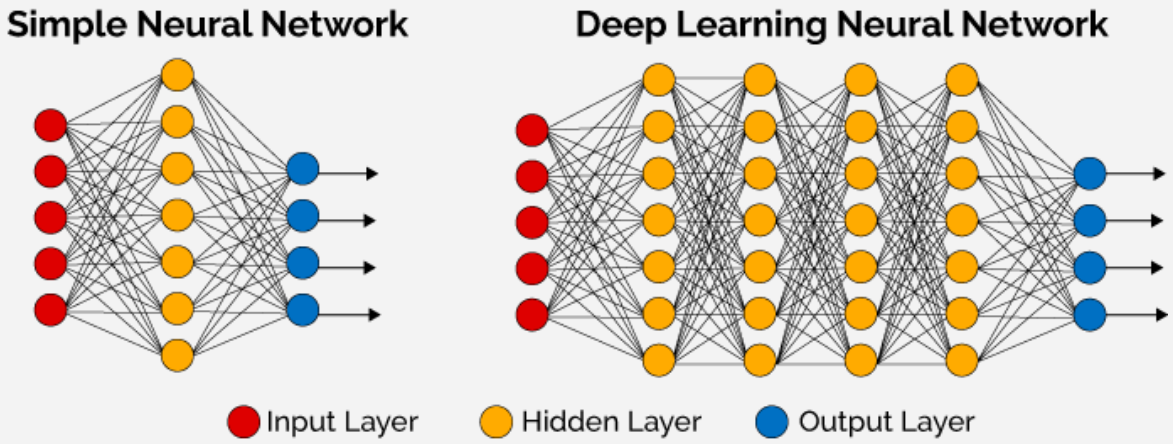


$$n = P_1W_1 + P_2W_2 + P_3W_3 + b$$
$$a = f(n)$$



Fonte da imagem: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg
Fonte da imagem: https://commons.wikimedia.org/wiki/File:Neuron_Hand-tuned.svg

O Que São Redes Neurais Artificiais Profundas ou Deep Learning?



Rede neural profunda
=
Rede feedforward com muitas camadas ocultas.

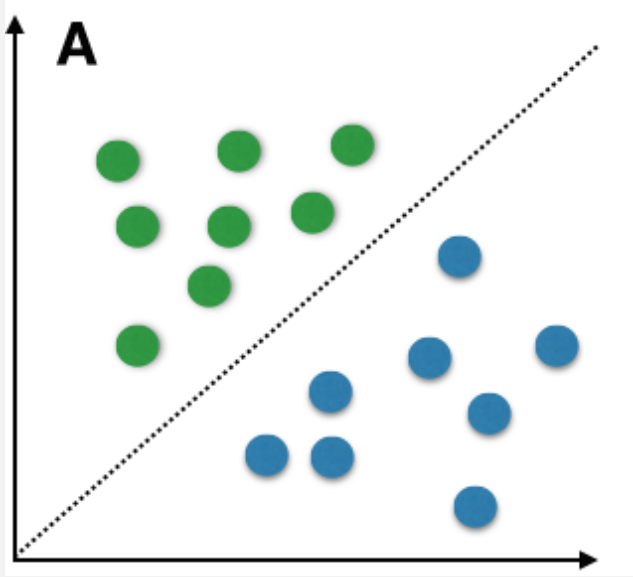
Quantas camadas uma rede deve ter para se qualificar como profunda?

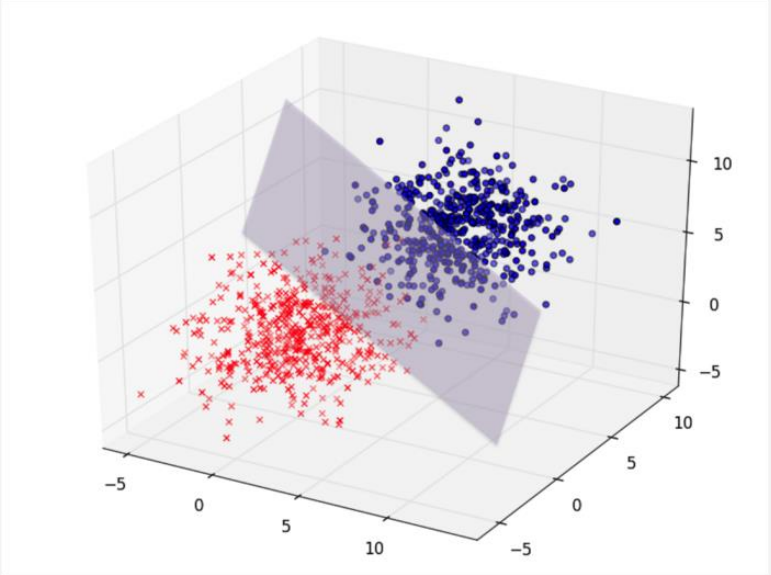
Arquiteura de Conexões

Arquitetura de Conexões

Deep Neural Networks Dimensões

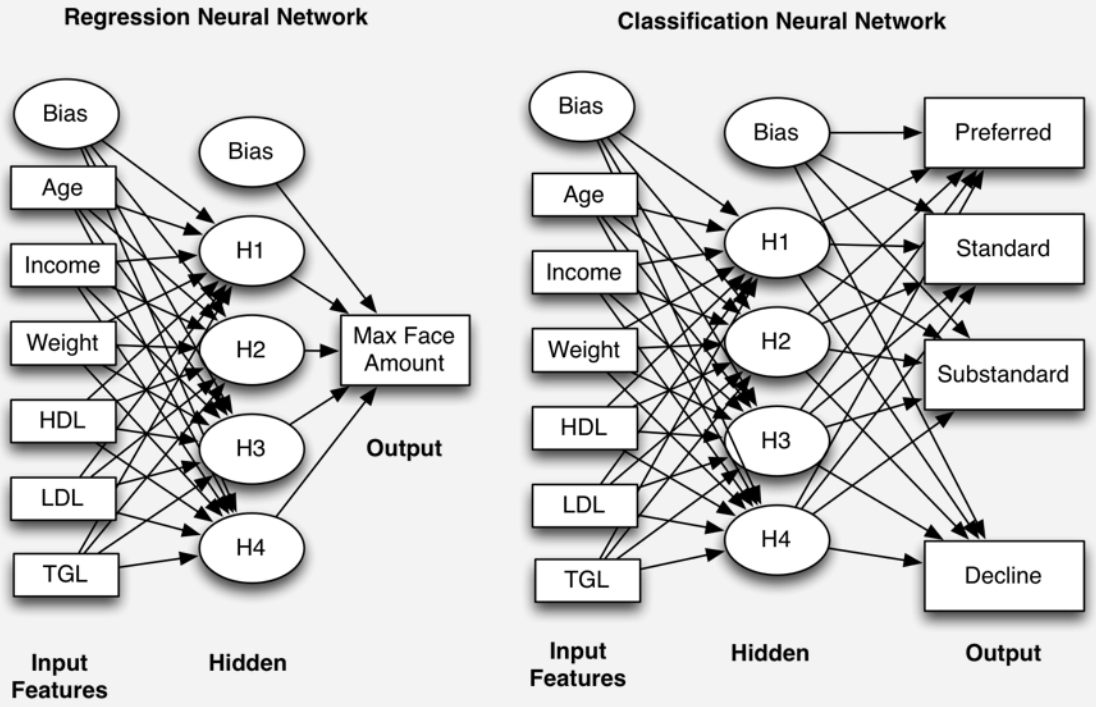
- **1D Vector** - Entrada clássica para uma rede neural, semelhante a linhas em uma planilha. Comum em modelagem preditiva
- **2D Matrix** - Entrada de imagem em escala de cinza
- **3D Matrix** - Entrada de imagem colorida
- **nD Matrix** - Entrada de ordem superior





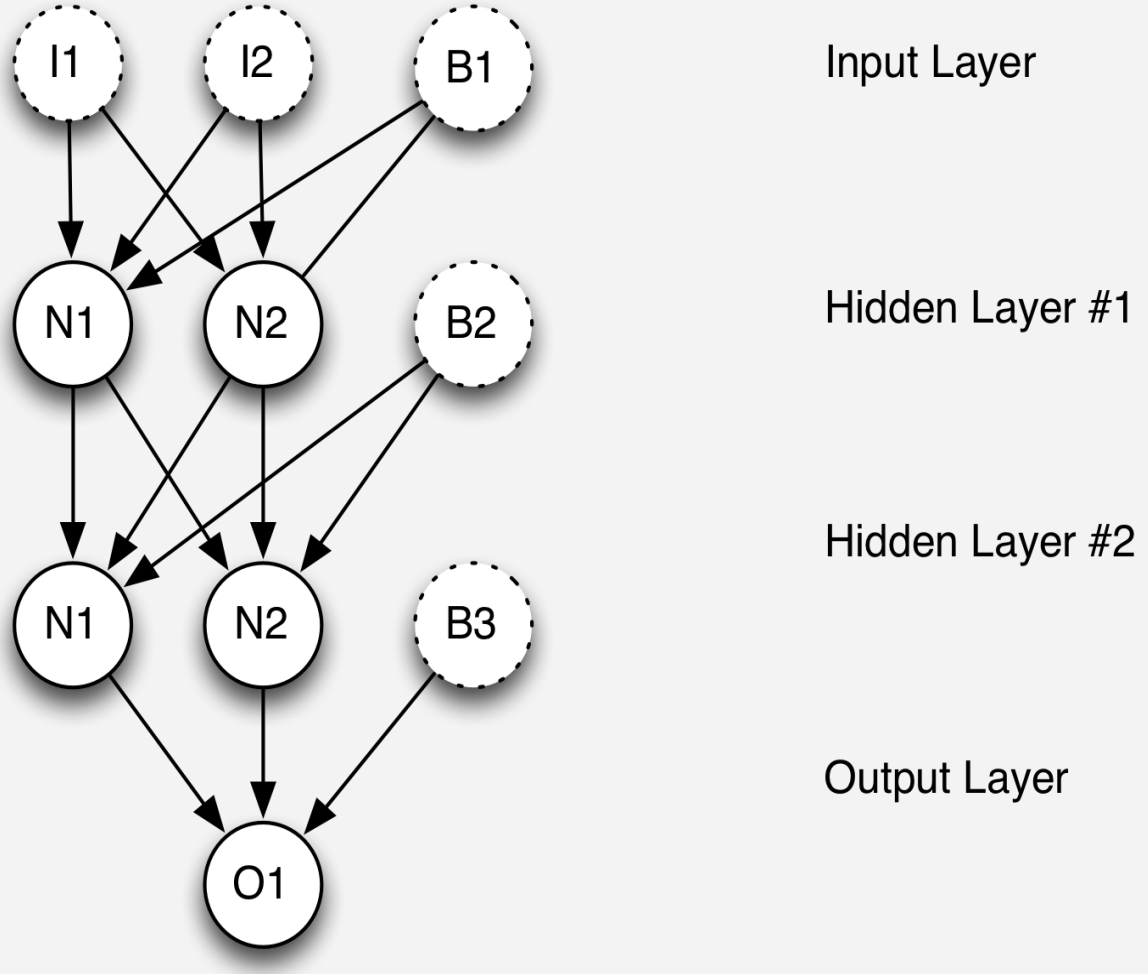
Deep Neural Networks
Classificação e Regressão

- Regressão - Você espera um número como sua previsão da rede neural.
- Classificação - Você espera uma classe / categoria como sua previsão da rede neural.



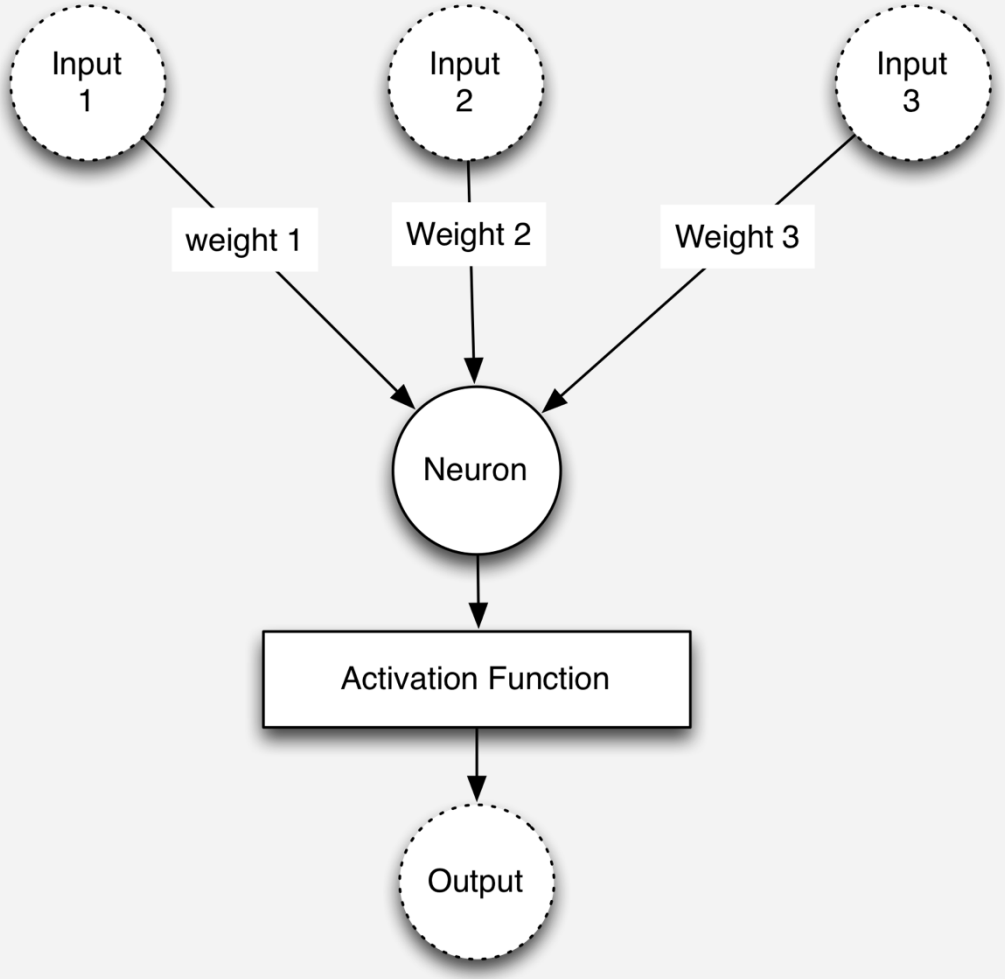
Uma rede neural típica

- **Input Layer**
  - A camada de entrada aceita vetores de recursos do conjunto de dados.
  - As camadas de entrada geralmente têm um neurônio de bias.
- **Output Layer**
  - O resultado da rede neural.
  - A camada de saída não possui um neurônio de bias.
- **Hidden Layers**
  - Camadas que ocorrem entre as camadas de entrada e saída.
  - Cada camada oculta geralmente terá um neurônio de bias.



Cálculo no Neurônio

$$f(x, \theta) = \phi(\sum_i (\theta_i \cdot x_i))$$



Funções de Ativação

A função ReLU é calculada da seguinte forma:

$$\phi(x) = \max(0, x)$$

O Softmax é calculado da seguinte forma:

$$\phi_i(z) = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}}$$

A função de ativação Softmax é útil apenas com mais de um neurônio de saída.
Mostra a probabilidade de cada uma das classes como sendo a escolha correta.

A função de ativação linear é essencialmente uma função de ativação usada para problemas de regressão:

$$\phi(x) = x$$

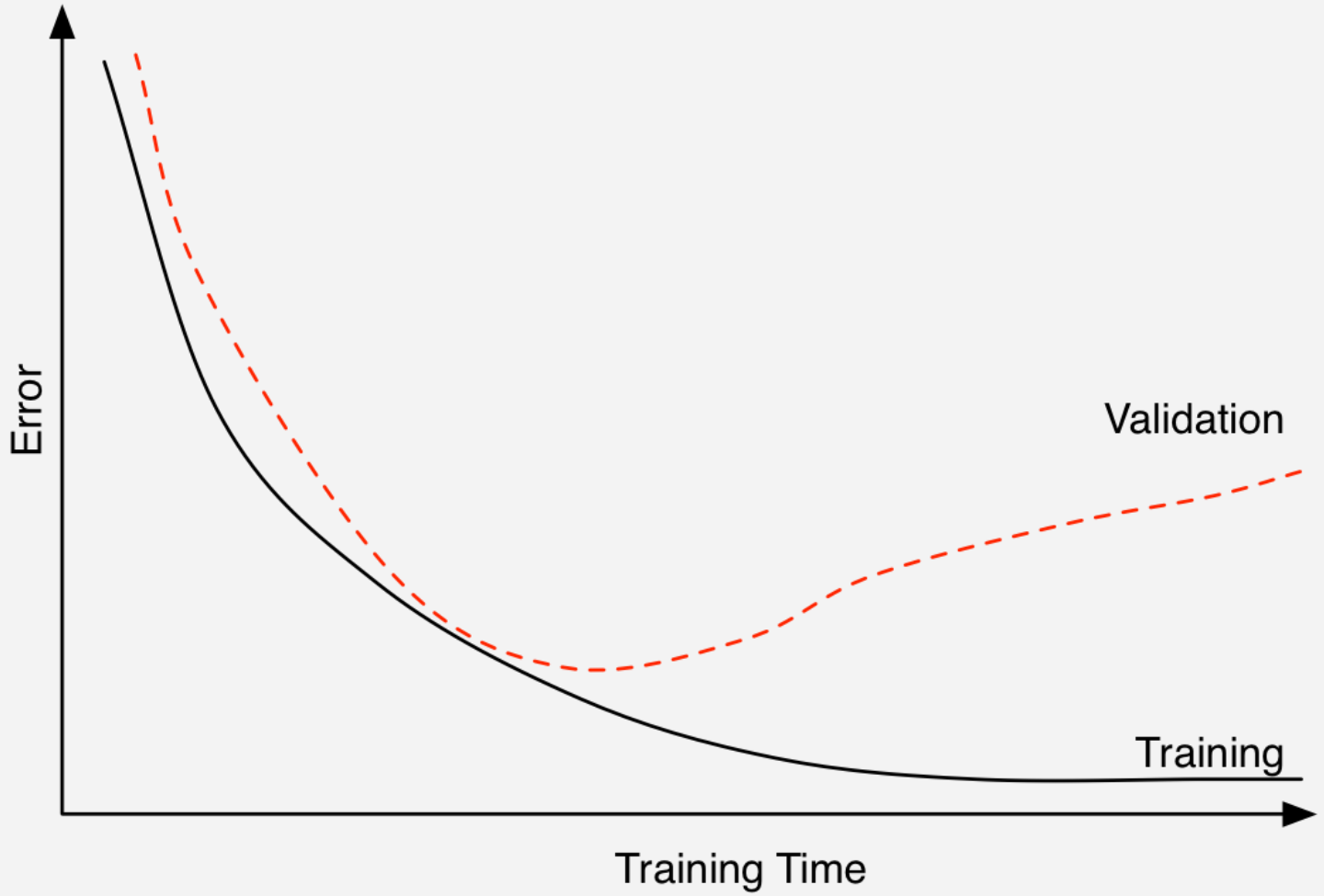
Normalização

- Uma transformação usual durante o treinamento de uma RNA é normalizar o conjunto de dados de treinamento de acordo com a distribuição normal padrão (i.e., média igual a zero e variância igual a 1) para evitar problemas de comparação devido às diferentes escalas usadas nos dados.

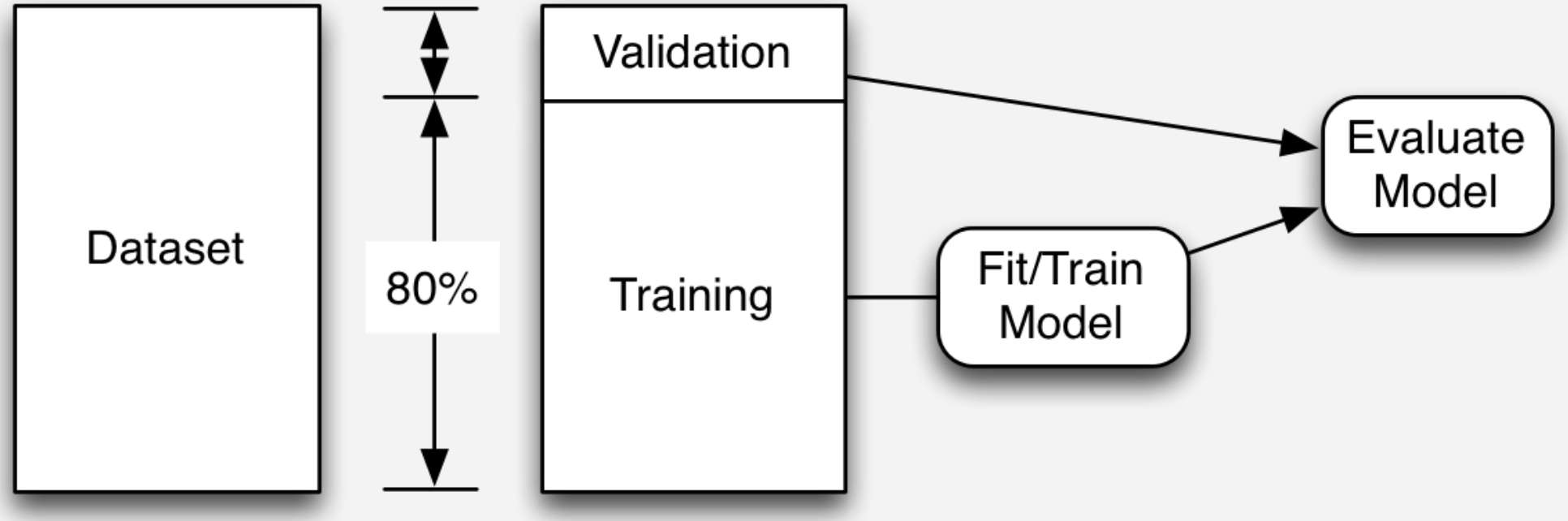
- A Normalização em Lote (Batch Normalization) é aplicada em cada mini-lote, para aumentar a eficiência durante a aplicação da transformação
- Tecnicas:
  - **One Hot Encoding**
  - Range
  - Escore Z

$$z = \frac{x - \mu}{\sigma}$$

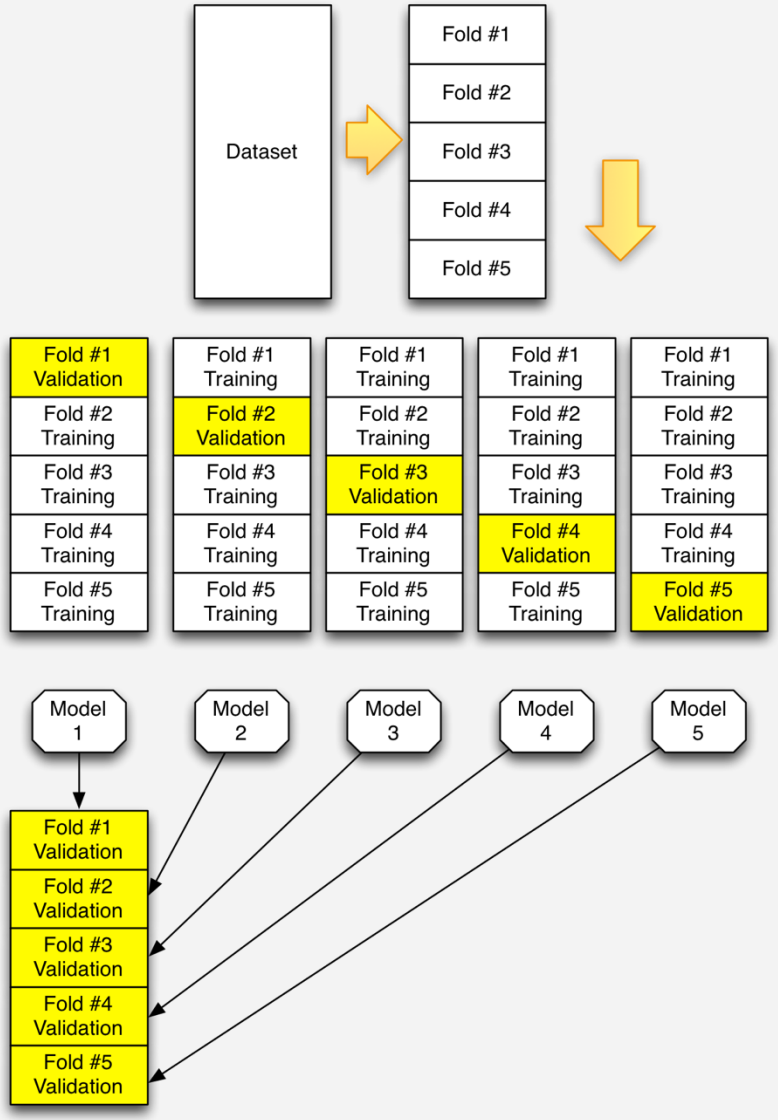
Conjunto de Validação e Early Stopping



Conjunto de Validação e Early Stopping



Treinamento com Cross Validation



Redes neurais convolucionais (CNN)

- Usado para visão computacional
- Carros autônomos, detecção de pedestres
- Em geral, melhor do que SVM (support vector machines)

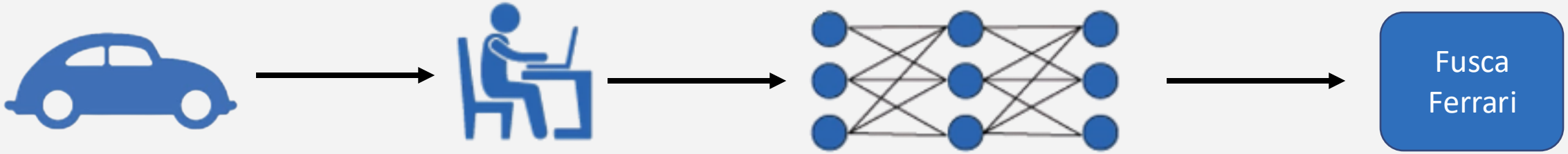
Aplicações

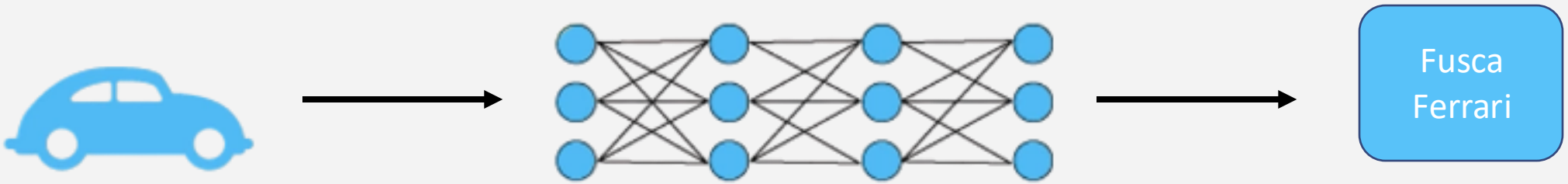
- WaveNet
- QuickDraw
- Convolutional Neural Networks (CNNs / ConvNets)

Aplicações

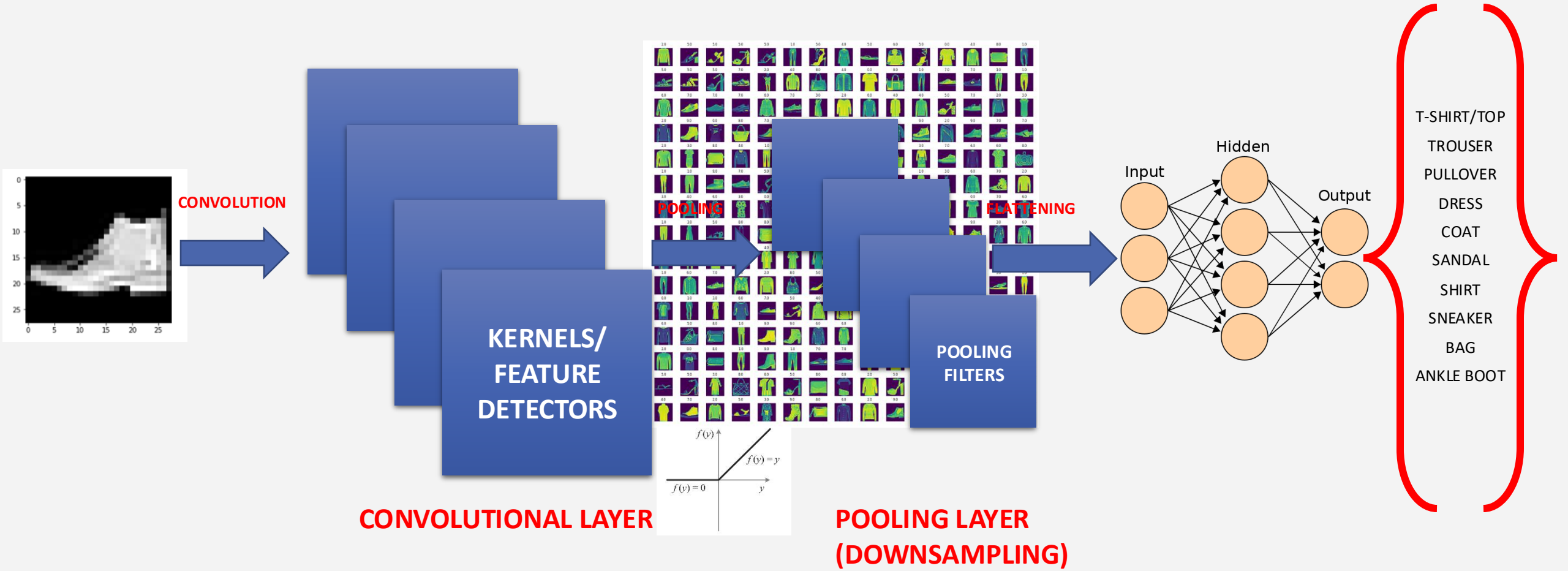
Aplicações

ML x DL





VISÃO GERAL DAS REDES NEURAIS CONVOLUCIONAIS



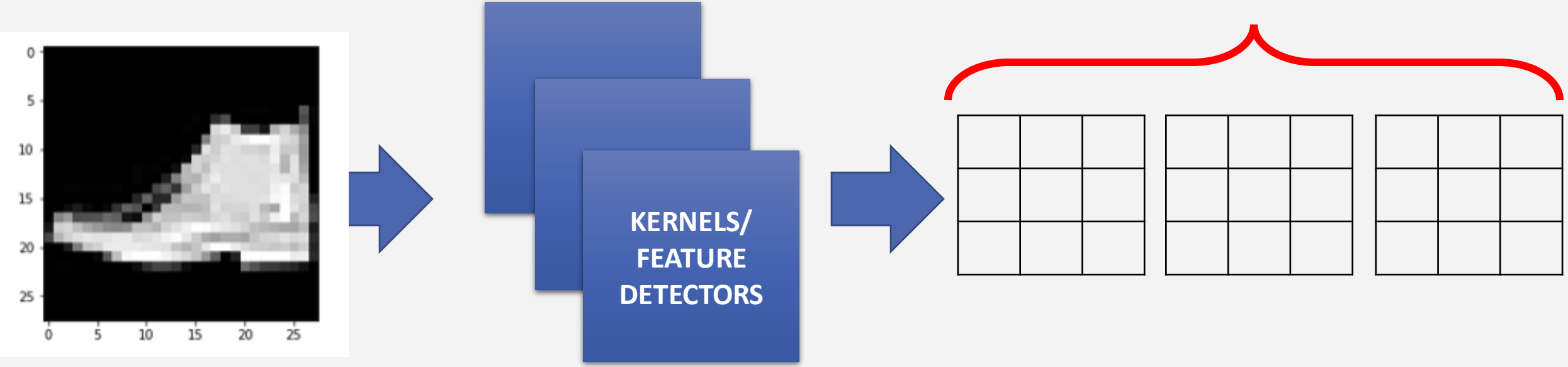
Fonte da imagem: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

Camadas de Convolução

DETECTORES DE CARACTERÍSTICAS

- As convoluções usam uma matriz para varrerem a imagem e aplicar um filtro para obter certo efeito
- Kernel é uma matriz para aplicar efeitos como embaçamento
- Selecionam as características mais importantes da imagem (pixels mais importantes)
- As convoluções preservam a relação espacial entre os pixels

FEATURE MAPS



DETECTORES DE CARACTERÍSTICAS

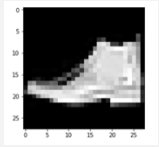
- **Exemplo on-line:** http://setosa.io/ev/image-kernels/

FEATURE DETECTOR

1	0	0
0	1	0
0	1	0

0	1	1	0	1
1	0	0	0	1
1	1	0	0	0
0	1	1	1	0
0	0	1	0	1

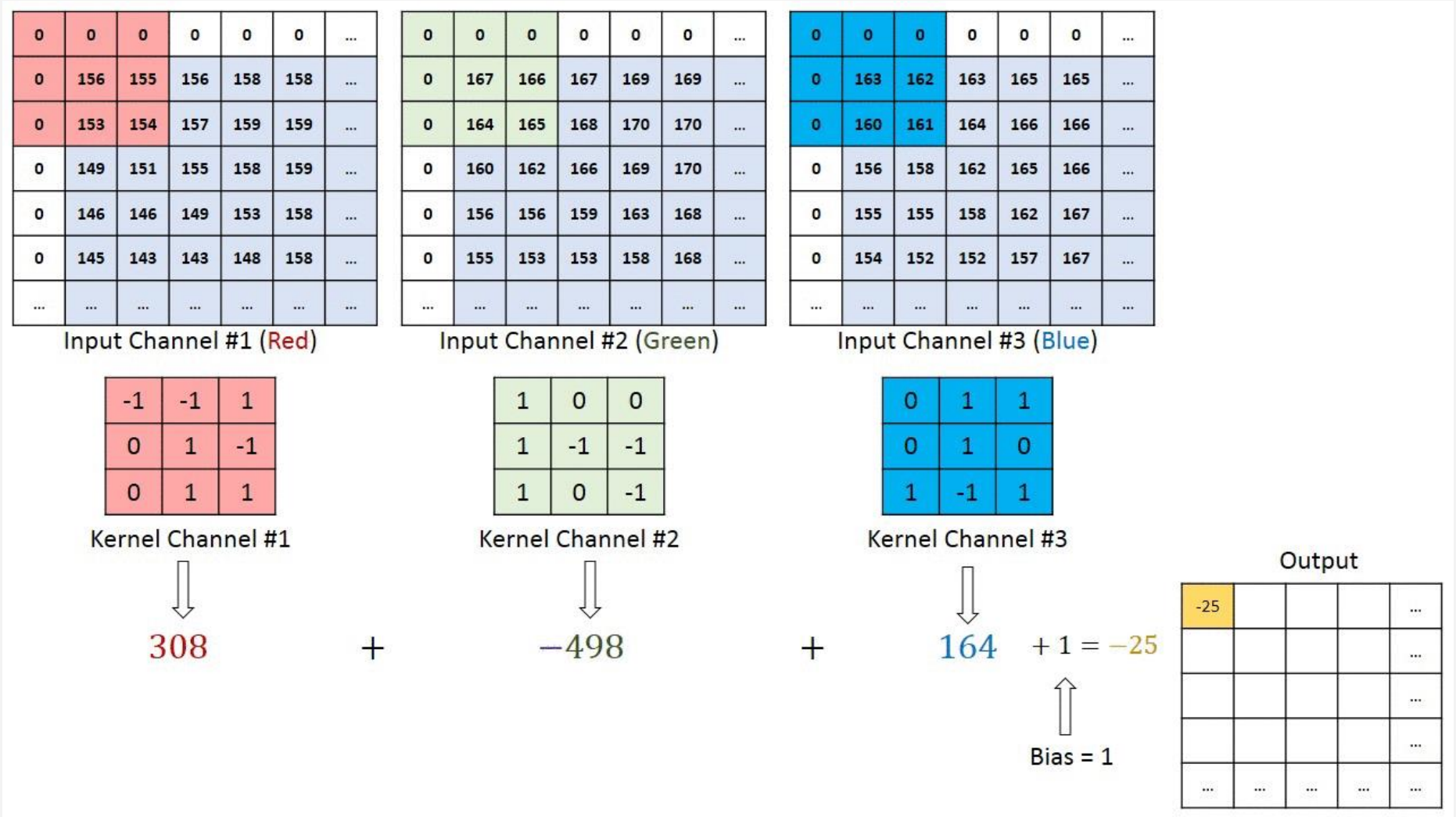
1	1	1
3	1	1
2	3	1



IMAGE

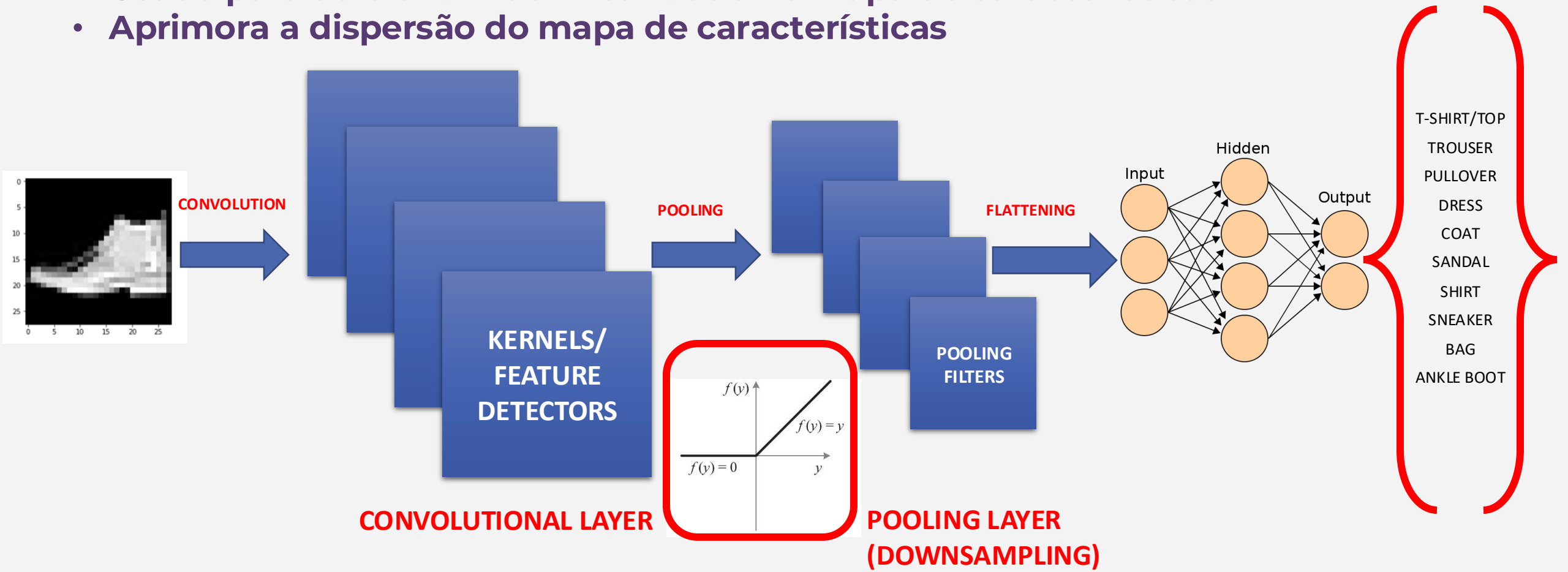
FEATURE MAP

Padding


Input Channel #1 (Red)
Input Channel #2 (Green)
Input Channel #3 (Blue)
Kernel Channel #1
Kernel Channel #2
Kernel Channel #3
308 + −498 + 164 + 1 = −25
Bias = 1
Output
-25

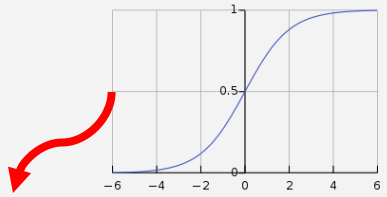
RELU (RECTIFIED LINEAR UNIT)

- Usado para adicionar não linearidade no mapa de características
- Aprimora a dispersão do mapa de características



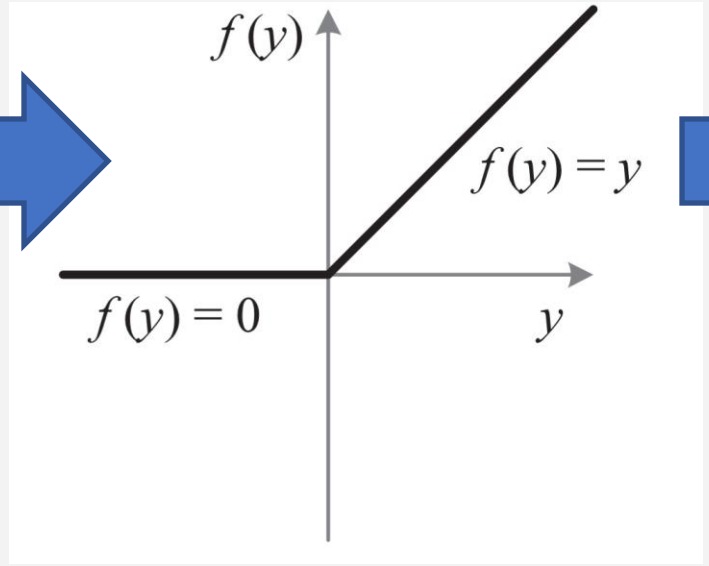
Fonte da imagem: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

RELU (RECTIFIED LINEAR UNITS)



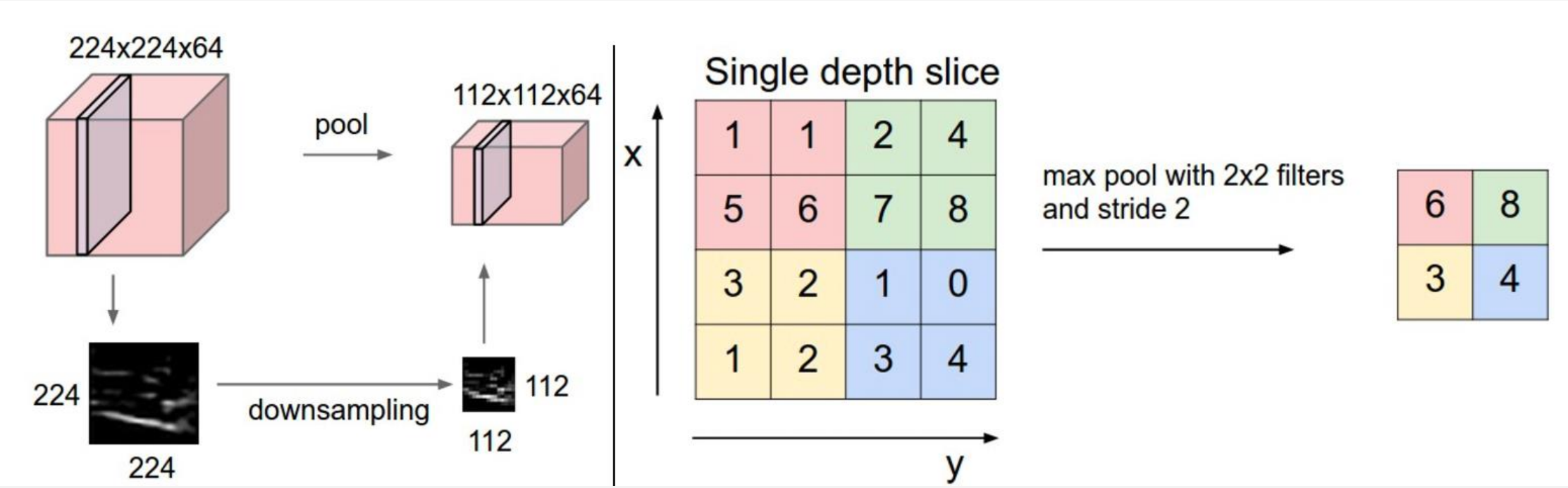
- O gradiente não desaparece se comparado com a função sigmoide

7	10	-5	2	1
1	0	2	3	-6
1	17	-5	0	0
0	1	1	1	0
0	0	-8	12	1



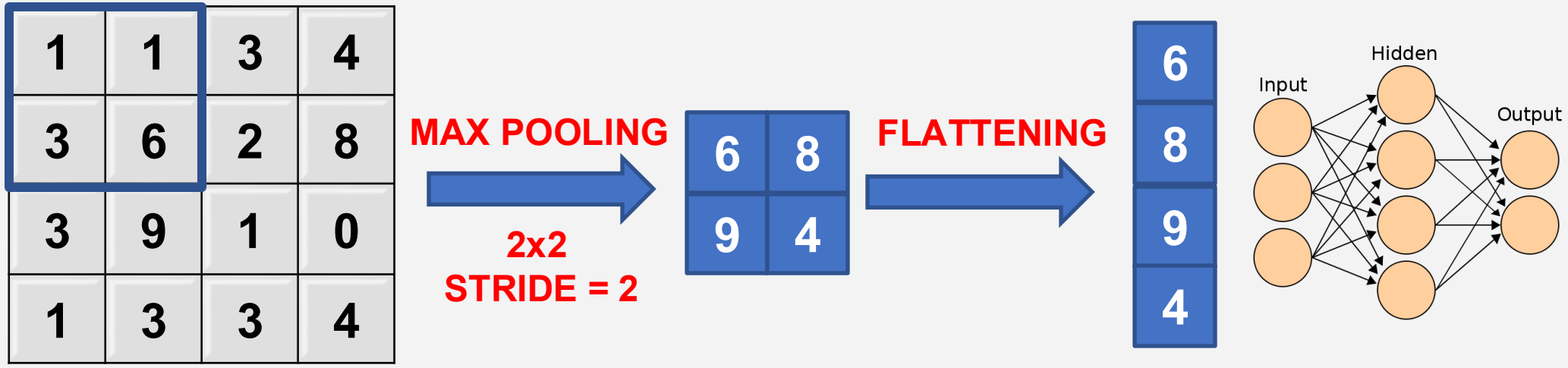
7	10	0	2	1
1	0	2	3	0
1	17	0	0	0
0	1	1	1	0
0	0	0	12	1

Camadas de Max Pooling



POOLING (DOWNSAMPLING)

- Reduz a dimensionalidade do mapa de características
- Aumenta a eficiência computacional, preservando as características
- Ajuda o modelo a generalizar melhor, prevenindo o overfitting
- Se um pixel muda de lugar, o mapa será o mesmo



ARQUITETURA LENET

- Arquitetura desenvolvida por Yann LeCun
- Artigo original: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- C: Convolution layer, S: subsampling layer, F: Fully Connected layer

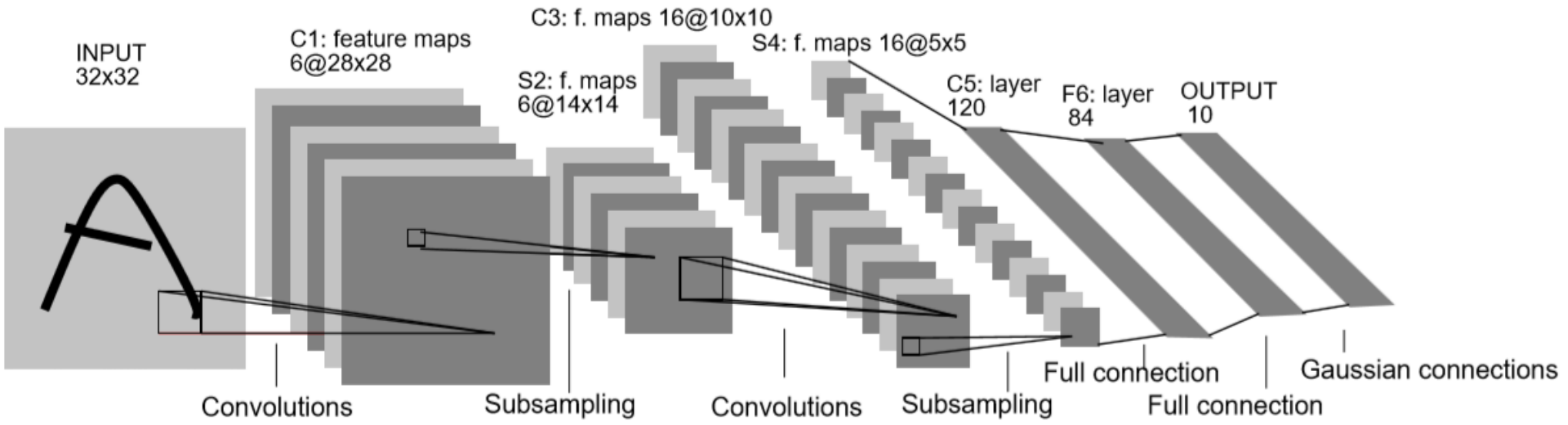


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

ARQUITETURA LENET

- **STEP 1: THE FIRST CONVOLUTIONAL LAYER #1**
  - Input = 32x32x1
  - Output = 28x28x6
  - Output = (Input-filter+1)/Stride* => (32-5+1)/1=28
  - Used a 5x5 Filter with input depth of 3 and output depth of 6
  - Apply a RELU Activation function to the output
  - pooling for input, Input = 28x28x6 and Output = 14x14x6

- **STEP 2: THE SECOND CONVOLUTIONAL LAYER #2**
  - Input = 14x14x6
  - Output = 10x10x16
  - Layer 2: Convolutional layer with Output = 10x10x16
  - Output = (Input-filter+1)/strides => 10 = 14-5+1/1
  - Apply a RELU Activation function to the output
  - Pooling with Input = 10x10x16 and Output = 5x5x16

- **STEP 3: FLATTENING THE NETWORK**
  - Flatten the network with Input = 5x5x16 and Output = 400

- **STEP 4: FULLY CONNECTED LAYER**
  - Layer 3: Fully Connected layer with Input = 400 and Output = 120
  - Apply a RELU Activation function to the output

- **STEP 5: ANOTHER FULLY CONNECTED LAYER**
  - Layer 4: Fully Connected Layer with Input = 120 and Output = 84
  - Apply a RELU Activation function to the output

- **STEP 6: FULLY CONNECTED LAYER**
  - Layer 5: Fully Connected layer with Input = 84 and Output = 43

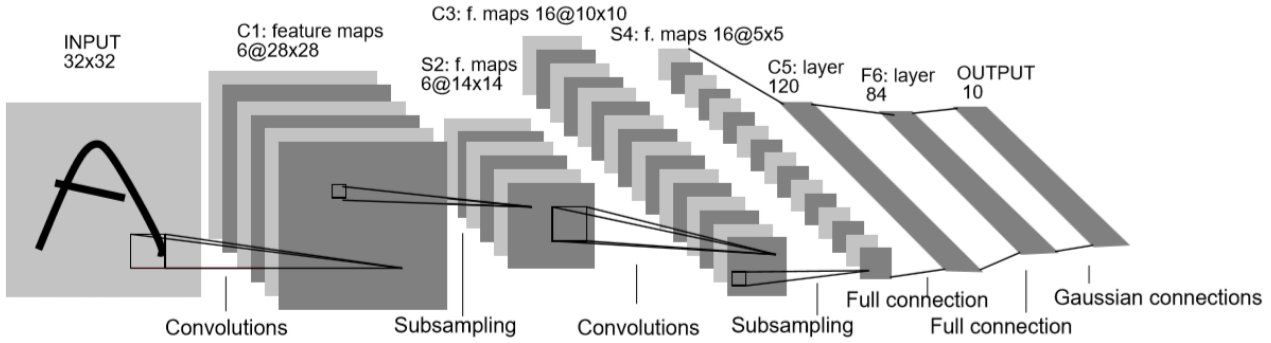


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

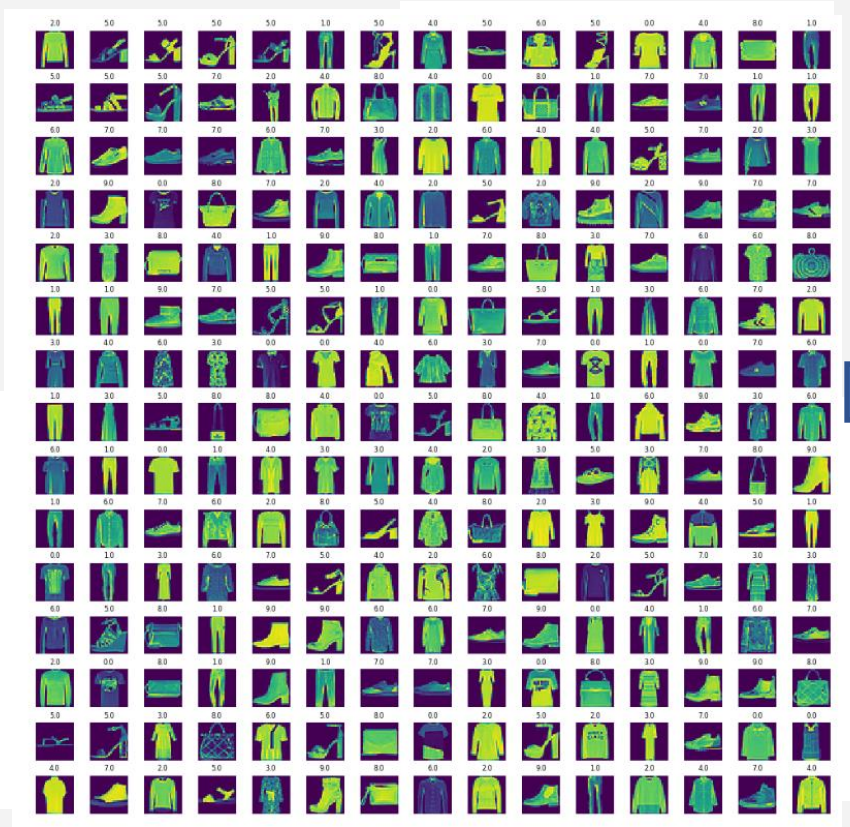
* Stride is the amount by which the kernel is shifted when the kernel is passed over the image.

FrameWorks

Framework	Plataforma	Escrito em	Interface em	Suporte ao CUDA	Suporte ao OpenCL	Suporte a RNN's	Suporte a CNN's	Processamento Paralelo
Tensorlow	Linux, MacOSx e Windows	C++ e Python	Python, C, C++	Sim	Em desenvolvimento	Sim	Sim	Sim
Theano	Diversas Plataformas	Python	Python	Sim	Em desenvolvimento	Sim	Sim	Sim
Caffe	Linux, MacOSx e Windows	C++	Python e Matlab	Sim	Em desenvolvimento	Sim	Sim	Parcial
Torch	Linux, MacOSx, Windows, Android, iOS	C, Lua	Lua, C, C++, PyTorch	Sim	Em desenvolvimento	Sim	Sim	Sim
Keras	Linux, MacOSx e Windows	Python	Python	Sim	Em desenvolvimento	Sim	Sim	Sim
CNTK	Windows, Linux, MacOSx (via docker)	C++	Python, C++ (.NET em breve)	Sim	Não	Sim	Sim	Sim
Deeplearning4j	Linux, MacOSx, Windows e Android	C, C++	Java, Scala, Clojure, Python (Keras)	Sim	Em desenvolvimento	Sim	Sim	Sim
MXNet	Linux, MacOSx, Windows, Android, iOS e Amazon AWS	C++	Python, C++, R, Matlab, Scala, Julia, Go, Perl e Java Script	Sim	Em desenvolvimento	Sim	Sim	Sim

PROJETO CLASSIFICAÇÃO DE ROUPAS

INPUT IMAGES



CLASSIFIER

Base de dados com 70.000 imagens
- 60.000 para treinamento
- 10.000 para teste

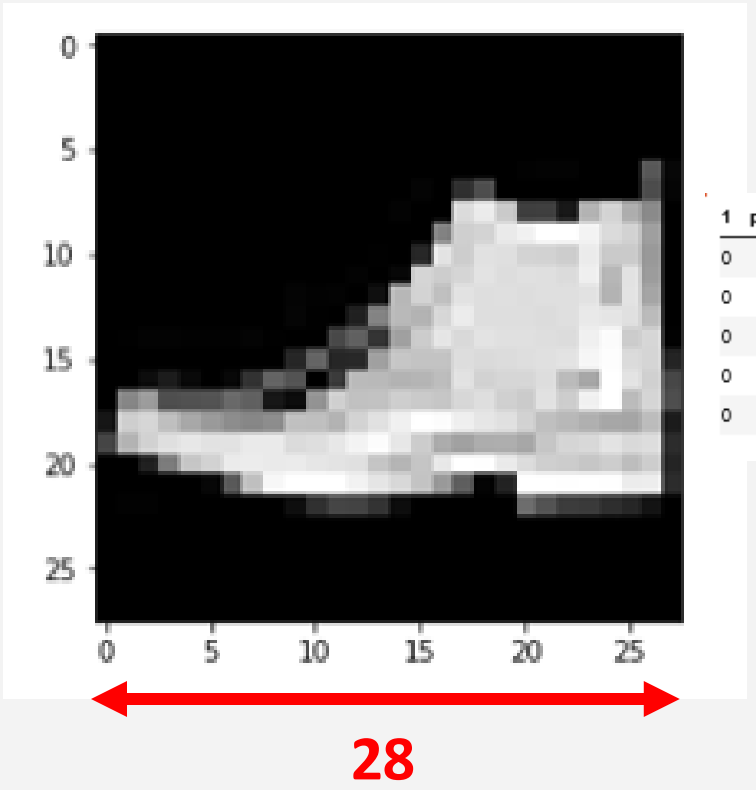
Imagens 28x28 em escala de cinza

TARGET CLASS: 10

- T-SHIRT/TOP
- TROUSER
- PULLOVER
- DRESS
- COAT
- SANDAL
- SHIRT
- SNEAKER
- BAG
- ANKLE BOOT

PROJETO CLASSIFICAÇÃO DE ROUPAS

- Imagens 28x28 em escala de cinza com valores na faixa de 0 até 255
- '0' representa o preto e '255' representa o branco
- Cada imagem é representada por uma linha com 784 posições



1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	5	0	...	0	0	0	30	43	0	0	0	0	0
0	0	0	1	2	0	0	0	0	...	3	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0


thank you
danke
謝謝
teşekkür ederim
gracias
obrigado
dziękuję
bedankt
go raibh maith agat
sukriya
kop khun krap
mochchakkeram
спасибо
tapadh leat
arigatô
dakujem
grazie
terima kasih
sagolun
takk
dank je
mauruuru
hvala
рахмат
Баярлалаа
mersi
barka
welalin
tack
spas
kia ora
vinaka
blagodaram
faafetai lava
misaotra
matondo
paldies
grazzi
mahalo
хвала
asante
manana
obrigada
murakoze
tenki
kiitos
dankie
nanni
nandri
dhanyavad
köszönöm
bayarlalaa
gràçie
enkosi
спасибі
akun dankon
dankon
djere dieuf
tau
дякую
mamnun
chokrane
chnorakaloutioun
grafias ago
gràcies
sulpáy
taiku
sobodi
dēkuji
mèsi
ありがとう
najis tuke
tanemirt
rahmet
trugarez
shukriya
мерси
dhanyavadagalu
diolch